비터비 알고리즘의 효율적인 연산을 위한 DSP 구조 설계

준회원 박 원 흠*, 종신회원 선우 명 훈**, 오 성 근**

Efficient DSP Architecture for Viterbi Algorithm

Weon heum Park*, Myung hoon Sunwoo**, Seong keun Oh** ***Regular Members***

요 약

본 논문은 다양한 무선 통신 표준에서 사용되는 비터비 알고리즘을 위한 전용의 DSP 명령어 및 하드웨어 구조를 제안한다. 제안한 구조는 비터비 알고리즘의 Trace Back(TB) 연산 사이클을 효과적으로 줄일 수 있다. 제안된 비터비 전용 명령어와 하드웨어 구조는 비터비 연산의 Add Compare Select(ACS) 연산 과정과 TB 연산 과정의 병렬 처리가 가능하며, 병렬 연산을 지원하기 위해 트랠리스 버터플라이 연산 과정에서 필요한 데이터를 자동으로 생성하는 Offset Calculation Unit(OCU)을 제안한다. 제안된 OCU는 삼성 SEC 0.18㎛ 라이브러리로 로직 합성하여 1,460 게이트 개수를 가지며, 최대 지연 시간은 5.75㎱를 나타내었다. 사용된 ACS-TB 병렬 처리 방식은 Eb/No 값이 6dB인 경우 MLSE 등화기 사용 사용되는 일반적인 TB 연산 방식과 비교하여 거의 동일한 BER 성능을 보여 주었으며, 제안한 DSP는 구속장 K=5 일 때 Carmel DSP와 비교하여 17%, TI TMS320c55x와 비교하여 45%의 연산 사이클이 줄일 수 있다.

Key Words : 비터비, DSP, 명령어, 무선통신, 에러정정

ABSTRACT

This paper presents specialized DSP instructions and their architecture for the Viterbi algorithm used in various wireless communication standards. The proposed architecture can significantly reduce the Trace Back (TB) latency. The proposed instructions perform the Add Compare Select (ACS) and TB operations in parallel and the architecture has special hardware, called the Offset Calculation Unit (OCU), which automatically calculates data addresses for the trellis butterfly computations. Logic synthesis has been performed using the Samsung SEC 0.18 ㎛ standard cell library. OCU consists of 1,460 gates and the maximum delay of OCU is about 5.75 ㎱. The BER performance of the ACS-TB parallel method increases about 0.00022dB at 6dB Eb/No compared with the typical TB method, which is negligible. When the constraint length K is 5, the proposed DSP architecture can reduce the decoding cycles about 17% compared with the Carmel DSP and about 45% compared with TMS320c55x.

Ⅰ. 서 론

길쌈 부호화는 GSM, GPRS, WCDMA, IS-95, CDMA2000 등 대부분의 무선 통신 표준을 위한 오류 정정 방식으로 널리 사용되어 왔다[1, 2]. GSM 시스템에서는 MLSE(Maximum Likelihood Sequen-

* 삼성전자 통신 연구소, **아주대학교 정보통신대학 전자공학부
논문번호 : KICS2004-11-281, 접수일자 : 2004년 11월 15일
※본 연구는 과학기술부에서 시행하는 국가지정연구실사업 시스템집적반도체기반기술개발사업 및IDEC의 부분적인 지원을 받아 수행되었습니다.

ce Estimation) 등화기를 위해 구속장 K=5인 비터비 복호기가 사용되며, 채널 복호를 위해서는 K=7인 길쌈 부호기가 사용된다. IS-95, WCDMA, CDMA 2000에서는 구속장 K=9인 비터비 복호기가 사용된다. 이와 같이 다양한 무선 통신 표준에서 사용되는 비터비 복호 알고리즘은 채널을 통하여 수신된 데이터들의 여러 경로를 탐색한 후, 그 중에 유사성(likelihood)이 가장 높은 경로를 선택하여 데이터를 복호하는 알고리즘이다 [3]. 이러한 비터비 알고리즘은 길쌈 부호(Convolutional code)의 MLD(maximum likelihood decoding)을 효율적으로 구현한 것이다 [4].

비터비 복호 과정은 일반적으로 전체 통신 시스템에서 많은 연산량을 필요로 한다 [5]. 그러므로 실시간 처리를 위해 주로 하드 와이어드 ASIC (Hardwired Application Specific Integrate Circuit)을 많이 사용하여 왔다 [5]. 그러나 프로그래머블 프로세서 기술과 성능이 빠르게 성장하면서 비터비 복호기의 구현은 다양한 통신 표준을 자유롭게 구현할 수 있는 프로그래머블 DSP로 점점 변화하고 있다 [5, 6]. 최근 개발된 저전력, 고성능의 상용 DSP들은 여러 가지 무선 통신 규격을 충분히 만족하도록 고성능을 가지고 있으며, 현재는 이러한 DSP 기술의 발전으로 SDR(Software Defined Radio) 플랫폼의 구현도 더욱 현실화 되고 있다 [6]. 대부분의 무선 통신을 위한 고성능 DSP들은 비터비 복호 연산을 위한 전용의 코프로세서 및 명령어를 지원한다 [1, 2, 7-10]. 그러나 비터비 연산을 위해 코프로세서를 사용할 경우 다양한 표준에 대한 고성능 처리는 가능하지만 추가적인 하드웨어에 대한 부담과 그에 따른 전력 소모가 커진다는 단점이 있다.

현재 비터비 복호 알고리즘을 하드웨어로 구현하는 방법은 주로 TB를 처리하는 방법에 따라 나누어진다 [7-8, 10-15]. 그 중에서 최근에 기존의 비터비 복호 알고리즘 대신 TB 연산 과정의 연산 사이클을 효과적으로 줄이는 하드 와이어드 비터비 복호기가 개발되었다 [13, 14]. 그러나 제안된 하드 와이어드 비터비 복호기는 하나의 칩으로 여러 가지 무선 통신 표준을 만족시키지 못한다는 단점이 있다. 본 논문에서 제안한 DSP는 비터비 복호 시에 TB 연산을 효율적으로 수행하여 총 연산 사이클을 줄이도록 DSP 명령어 및 하드웨어 구조를 제안한다. 또한, 제안한 DSP 명령어 와 하드웨어 구조는 다양한 통신 표준을 만족하도록 설계되어 하드 와이어드 비터비 복호기의 단점을 극복하였다.

본 논문은 다음과 같이 구성된다. 2장에서는 제안한 DSP에 적용되는 비터비 복호 알고리즘에 대해 설명하고, 3장에서 현재 상용화된 비터비 복호를 위한 DSP 구조와 전용 명령어에 대해 서술한다. 4장에서는 제안된 비터비 전용의 DSP 명령어 집합과 하드웨어 구조에 대해 구체적으로 설명하고, 5장에서는 상용 DSP와 제안된 DSP의 성능을 비교하며 마지막으로 6장에서 결론을 맺는다.

Ⅱ. 비터비 복호 알고리즘

비터비 복호 과정은 일반적으로 가지 매트릭(Branch Metric, BM) 연산 과정, ACS 연산 과정, 생존자 경로 연산 과정의 단계로 크게 진행 된다. 각 과정은 다음과 같다. 가지 매트릭 연산 과정은 트랠리스 도에서 얻은 코드 워드와 수신된 데이터 간의 거리 값을 구하는 과정이다. 가지 매트릭은 입력 데이터가 전송할 때 생기는 오류의 크기를 나타내며 가지 연산 과정은 가지 매트릭 유닛(BMU)에서 이루어진다. 다음의 ACS 연산 과정은 데이터를 비교하여 오류가 가장 적은 경로를 선택하는 과정이다. 즉, 이전 상태까지 누적된 경로 매트릭과 수신 부호의 가지 매트릭 값을 사용하여 각 상태로 천이되는 경로를 비교하여 경로 매트릭 값이 적은 경로를 선택한다. 다음 과정인 생존자 경로 연산 과정에서는 새롭게 선택되어진 생존자 경로 값을 생존자 경로 메모리에 저장하고 갱신하게 된다.

그림 1은 비터비 버터플라이의 구조를 나타낸다. 트랠리스도의 현재 상태(State)에서 이전 상태의 경로 매트릭 값과 그에 대응하는 가지 매트릭 값이 더해지고, 이렇게 계산된 새로운 경로 매트릭 값은 다른 모든 가능한 경로와 비교한 뒤 최소 값으로 결정된다.

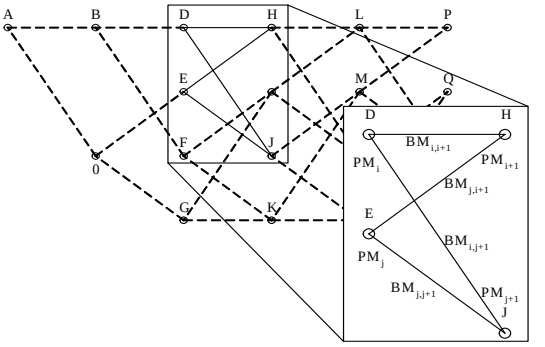


그림 1. 비터비 버터플라이 구조(K=3, rate=1/2)

ACS 연산 과정은 다음 상태(Next state)의 경로 매트릭 값을 계산하는 과정이며, 다음의 식 (1)에 따라 계산된다.

$$PM_{i+1} = \min(PM_i + BM_{i,i+1},\ PM_j + BM_{j,i+1})$$
$$PM_{j+1} = \min(PM_i + BM_{i,j+1},\ PM_j + BM_{j,j+1}) \quad (1)$$

여기서, 부호율이 1/n 인 비터비 시스템은 트랠리스 구조에서 가지 매트릭 값이 대칭적으로 나타나게 되므로, 가지 매트릭 값이 아래의 식 (2)에서와 같이 간략화 된다.

$$BM_{i,i+1} = BM_{j,j+1},\ BM_{i,j+1} = BM_{j,i+1} \quad (2)$$

ACS 연산 과정에서 얻어진 경로 선택 정보는 시스템의 역추적 깊이(D)만큼 저장된 후, 최소 경로 매트릭을 갖게 되는 상태를 초기값으로 하여 역추적 메모리에 저장된 정보를 이용해 데이터를 복호하게 된다 [15].

ACS 연산 과정에서 생존자 경로를 구현하는 방법은 크게 두 가지로 나누어진다. 그중 하나는 연산 지연 시간이 적은 레지스터 교환 방식이다. 이 방법은 연산의 지연 시간이 적은 반면 칩 면적이 크며 높은 전력 소모를 요구한다는 단점이 있다. 또 다른 방법은 TB를 이용한 방법이다. 이 방법은 레지스터 교환 방식보다는 좀더 나은 성능을 보이지만 비교적 출력 지연 시간이 크다는 단점이 있다[15].

이러한 방법 외에 ACS 연산 시에 TB 연산을 처리하는 병렬 처리 방법이 있다 [13, 14]. Trace-Back Free 비터비 복호기는 이러한 ACS-TB 병렬 처리 방법을 사용하는 구조를 가지며, 패스트 포인터 조정 회로(Fast pointer adjustment logic)를 사용하여 생존자 경로의 변화를 추적하며 ACS 연산 과정 중에 병렬적으로 생존자 경로를 저장하게 된다[13]. 이 방식은 전력 소모가 적은 반면에 하드웨어 구현 시 그 복잡도가 큰 단점이 있다. 다른 방법은 TB 연산 시에 사용되는 메모리의 용량을 줄일 수 있는 Look-ahead TB 비터비 복호 방법이다 [14]. Look-Ahead TB 방법에서는 디코딩 역추적 깊이 이후에 TB 연산을 하지 않고, 각 스테이지마다 TB 연산을 수행한다 [14]. 이 방법을 사용할 경우, TB 제어 유닛이 복잡하지 않으며 TB 연산 과정에서 필요한 메모리의 크기가 줄어드는 장점이 있다[14].

현재 ACS 계산 과정과 TB 연산 과정을 병렬적으로 수행하는 비터비 복호기는 하드 와이어드ASIC으로 구현되어 왔다 [13, 14]. 그러나 이 같은 비터비 복호기는 DSP처럼 프로그래머블 하지 않기 때문에 여러 가지 종류의 무선 통신 표준을 하나의 칩으로 만족하기가 어렵다.

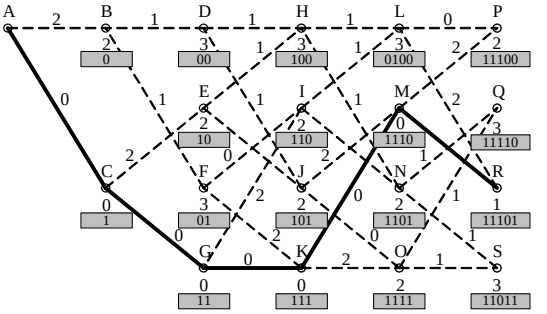


그림 2. ACS-TB 병렬 처리에 의한 비터비 복호 방법

그림 2는 TB의 지연을 줄이는 비터비 복호 과정을 간략히 나타낸다. 그림에 나타난 점선은 비터비 복호 시에 선택되지 않은 경로를 나타내고, 굵은 직선은 선택된 생존자 경로를 나타낸다. 또, 회색 칸의 숫자는 스테이지마다 계산된 디코딩 값이 데이터 메모리 안에 저장되는 과정을 나타낸다. 이와 같이 ACS-TB 병렬 처리 방법에 의한 하드웨어를 본 논문에서 제안한 DSP에 적용하기에 앞서, ACS-TB 병렬 처리 방법과 일반적인 TB 방법과의 BER 성능 비교를 실시하였다. 시뮬레이션에 사용된 MLSE 등화기는 채널 추정기로 상관기 방식을 이용하였고, 비터비 연산은 구속장 K=5, 디코딩 깊이는 29인 비터비 프로세서를 사용하였다.

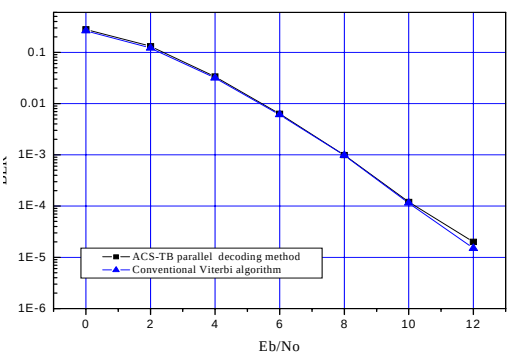


그림 3. 비터비 복호 방법에 의한 성능 비교

그림 3에서 보는바와 같이 시뮬레이션 결과 Eb/No 값이 6dB 일 때, 일반적인 TB 방식의 연산

은 0.00609dB, ACS-TB 병렬 처리 비터비 복호 방식은 0.00631dB로 약 0.00022dB정도 높은게 측정되었다. 하지만 이 값은 일반적인 TB 연산 방식의 BER 성능과 비교할 때 무시할 수 있는 수준이다. 다시 말해서 BER 성능 시뮬레이션 결과, 일반적인 TB 연산 방식과 ACS-TB 병렬 처리 방식의 전체적인 BER 성능은 거의 차이가 나지 않았다. 이 같은 이유로 ACS-TB 연산을 병렬적으로 처리해도 비터비 복호기의 성능에는 거의 영향을 미치지 않는다는 것을 알 수 있다.

ACS-TB 병렬 처리 연산 방법은 비터비 복호기의 BER 손실이 일반적인 TB 방법에 비해 거의 없으면서 고성능 비터비 연산이 가능하다 뿐만 아니라 하드웨어 복잡도를 줄일 수 있으며, TB 연산 시에 사용되는 메모리 사용량을 줄일 수 있다[13, 14]. 전통적인 비터비 복호기는 ACS 연산 과정이 지난 다음에 현재 상태값에 대한 경로 선택 정보(Selection bit)와 경로 매트릭 값을 저장하게 된다. 하지만 제안된 DSP는 경로 매트릭 값과 경로 선택 정보 값을 ACS-TB 병렬 처리 방법에 의해 병렬적으로 갱신하게 된다.

Ⅲ. 비터비 복호를 위한 상용 DSP

비터비 복호기를 ASIC으로 구현할 경우에 여러 가지 무선 통신 표준을 하나의 칩에서 모두 구현하기가 어려우며, 무선 통신 SDR 플랫폼에도 적용되기 어렵다. 이 때문에, 무선 통신 시스템이나 통신 SDR 플랫폼에서는 전용의 하드웨어나 명령어를 가진 DSP 구조가 적합하다. 최근 비터비 연산을 지원하는 상용 DSP 들은 비터비 알고리즘에서 많은 연산량을 차지하는 ACS 연산을 위해 데이터 처리 유닛(Data Processing Unit, DPU) 내부에 여러 개의 연산기 및 비교기를 가지고 있으며, TB 연산을 위해서 전용의 레지스터를 가지고 있는 경우가 많다[1, 2, 7-10].

상용 DSP에서 사용되는 비터비 전용 명령어는 다음과 같다. StarCore SC140은 ACS 연산을 위해 ADD2, SUB2, MAX2BIT을 지원하고, 생존자 경로 계산을 위해 VSL.4W와 VSL.4L 명령어를 지원한다[1]. Carmel DSP는 ACS 연산을 위해 ADD2, SUB2, MAXBTR 명령어가 사용 가능하다[2]. CVP는 가지 매트릭 계산 과정을 위해 ACS, MANH 명령어를 사용한다[8]. TI TMS320c55x는 ACS 연산을 위해 ADDSUB, SUBADD, MAXDIFF 명령어를 사용한다[9]. ZSP500은 ACS 연산을 위해서 VIT_A, VIT_B 명령어를 지원한다[10].

위에서 언급된 비터비 전용 DSP 중, StarCore SC140과 CVP는 보다 많은 연산 유닛을 가지는 VLIW(Very Long Instruction Word) 구조의 코어이며, ZSP500은 슈퍼스칼라(Superscalar) 코어이므로 비터비 복호시의 성능이 전반적으로 우수하다. 또한, ASDSP의 구조는 슬라이스라는 단위 셀로 구성되어 있고, 슬라이스는 ALU와 곱셈기, 배럴 쉬프터, 레지스터 셋으로 구성된다[8].

이와 같이 일반적으로 디지털 무선 통신 처리를 목적으로 하는 상용 DSP들은 여러 가지 통신 시스템에 범용으로 적용 가능하도록 많은 연산 유닛을 가지는 VLIW나 슈퍼스칼라 구조를 가진다. 그러나 이러한 코어들은 전력 소모나 칩 면적 등의 이유로 모바일 응용에서는 사용되기 힘들다.

비터비 복호 알고리즘을 DSP로 처리할 경우 ACS 연산 과정이 비교적 많은 연산량을 가지게 되며 TB 연산량은 구속장의 길이가 증가할수록 상대적으로 급격히 증가하게 된다. 일반적으로 비터비 알고리즘을 상용 DSP로 구현할 경우 대부분의 DSP가 ACS 연산 과정 이후에 TB 연산을 처리하게 된다[1, 2, 7-10]. ACS-TB 병렬 처리를 위해서는 DSP가 많은 양의 데이터를 메모리로부터 읽고 쓰면서 동시에 ACS 연산과 TB 연산을 수행할 수 있어야 하기 때문이다. 현재 상용 DSP는 이와 같은 ACS-TB 병렬 처리를 지원하는 명령어나 특별한 하드웨어를 가지고 있지 않으므로, ACS-TB 병렬 처리 연산을 수행할 수 없다. 그러나 본 논문에서 제안된 DSP는 적은 하드웨어 크기로 ACS-TB 병렬 처리를 지원하도록 설계 되었다.

Ⅳ. 비터비 복호를 위해 제안된 전용의 명령어 및 하드웨어 구조

본 절에서는 비터비 복호의 효율적인 연산을 위해 제안된 전용의 명령어 및 하드웨어 구조에 대해 기술한다.

4.1 비터비 복호를 위해 제안된 전용 명령어 집합

효율적인 비터비 복호를 위해 제안된 명령어 집합은 다음과 같다. 비터비 연산을 실행하기 전에 구속장 및 부호율 값을 초기화 할 때 사용되는 VS(Viterbi Setting) 명령어, ACS 및 TB 연산의 병렬 처리를 지원하는 ACSP(Add Compare Select

표 1. 비터비 전용 명령어

Inst. / Name	문법	동작 내용
VS	vs #N	V-Flag←set, CSR←#N, CDR←#N/2
ACSP	acsp GR_x.E, GR_y.E, GR_{tmp}	min [(GR_x+GR_{x+1}),(GR_y+GR_{y+1})]; if [(GR_x+GR_{x+1})>(GR_y+GR_{y+1})], GR_{tmp}=1, else GR_{tmp}=0;
TBP	tbp GR_{tmp1}, GR_{tmp2}	if [GRtmp1=1], GR_{tmp2}=GR_{tmp2}<<1|0x0001 else GR_{tmp2}=GR_{tmp2}<<1;

#N : The total number of states, i.e., 2^{K-1}

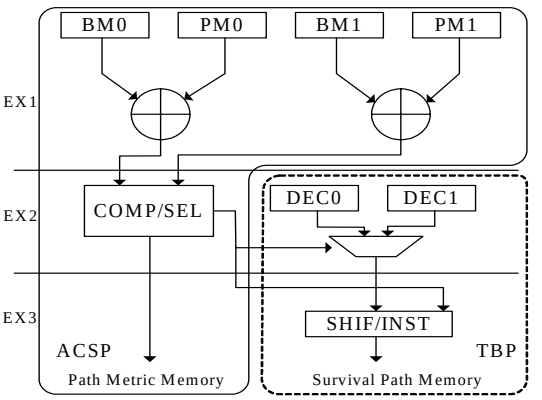
Parallel) 명령어, TBP(Trace Back Parallel) 명령어를 제안한다.

표 1은 비터비 복호를 위해 제안된 명령어의 구문과 동작 내용을 나타낸다. ACS-TB 연산의 병렬 처리 과정은 전용의 하드웨어 지원으로 가능하므로 표 1에 나타난 명령어의 동작 내용에는 포함시키지 않았다. VS 명령어는 DSP의 상태 레지스터(Status Register)에 비터비 동작을 알리는 비터비 플래그를 세팅하며, OCU 내의 전용 레지스터에 비터비 연산의 상태 값을 입력하는데 사용된다. 즉, VS 명령어는 OCU 내의 CSR, CDR 레지스터에 구속장 값과 구속장 값을 2로 나눈 데이터를 각각 입력하게 된다. 초기값을 입력하면 OCU가 Load, Store 등의 메모리 읽기, 쓰기 명령어 사용 없이 비터비 복호에 필요한 메모리 주소의 오프셋 값을 자동으로 생성하게 된다. 그러므로 OCU에 의해 제안된 새로운 주소 생성 방식은 비터비 연산 시에 데이터 이동에 의한 사이클을 줄이는 역할을 한다.

ACSP 명령어는 ACS 버터플라이 연산 및 비교 연산을 수행한 뒤에 디코딩된 경로 선택 정보 비트를 출력하는 명령어이다. TBP 명령어는 ACS-TB 병렬 연산을 수행할 때 사용되며 ACS 연산과 동시에 TB 연산을 할 수 있도록 DPU 내에 데이터 경로를 지원한다. 제안된 DSP는 ACSP 명령어와 TBP 명령어를 동시에 사용하여 ACS 연산과 TB 연산을 수행하며, 그림 4는 ACSP 연산과 TB 연산의 병렬 처리 과정을 데이터 경로의 입장에서 나타낸다.

그림 4에 나타난 데이터 흐름은 다음과 같다.

- Excute 1(EX1) 스테이지에서 덧셈기 블록은 가지 매트릭 값과 경로 매트릭 값을 더한다.
- EX2 스테이지에서 COMP/SEL(Compare/Select) 블록은 이전 스테이지에서 계산된 값 중에 작은 값을 비교, 선택하고 경로 선택 정보 비트(Selection Bit)는 병렬적으로 경로 결정 값(Decision value)을 선택하게 된다.



DECs : The decision values of previous states

그림 4. ACSP, TBP 명령어의 연산 흐름

- EX3 스테이지에서 SHIF/INST(Shift and Insert) 블록은 선택된 결정 값을 레지스터의 LSB부터 채워 넣는다.

이와 같은 방법으로 ACS-TB 병렬 처리에 의해 계산된 경로 선택 정보 비트는 경로 매트릭 메모리 구역에 저장되고, 경로 결정 값들은 메모리 구조에 맞게 변경되어 생존 경로 메모리 구역에 저장된다. 이 메모리 주소값은 OCU 블록에서 ACS-TB 연산 시 동시에 생성되며, OCU가 생성하는 메모리 주소의 구성도는 그림 5와 같다. OCU의 동작으로 데이터 이동을 위한 추가적인 지연 시간이 없어지므로, 전체 비터비 복호 시스템의 어셈블리 프로그램 코드

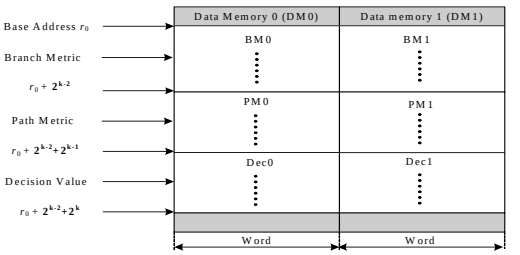


그림 5. 비터비 복호를 위한 데이터 메모리 구성도

길이 또한 짧아지게 되어 프로그램 메모리 접근에 의한 전력 소모를 줄일 수 있으며, 사용자가 소프트웨어적으로 주소를 생성해야 하는 부담 또한 줄일 수 있다.

4.2 제안된 전체 DSP 구조

현재 상용 DSP는 비터비 복호 연산 시에 가지 매트릭 값, 경로 매트릭 값, 경로 결정값(Decision value) 들을 위해 사용자가 적절히 주소값을 구성하여 주어야 한다 [1, 2, 9]. 그러나 제안된 DSP는 데이터의 입, 출력을 위한 베이스(Base) 주소값 만을 지정하여 주면 ACS, TB 연산 시에 필요한 주소를 OCU에서 자동으로 생성하여 전체 연산을 효율적으로 수행하게 된다. 이와 같이 새로운 주소 생성 모드를 지원하는 OCU는 DSP의 주소 생성 유닛(Address Generation Unit, AGU) 안에 적은 게이트 수로 구성되어 있어 효율적이다. 즉, 비터비 연산 과정에서 데이터 연산 유닛(DPU)이 ACS-TB 연산을 수행할 때, OCU는 필요한 주소를 생성하기 때문에 버터플라이 연산 시 추가적으로 필요한 프로그램 코드가 제거되고 데이터 이동과 연산이 병렬로 처리되므로 비터비 복호의 가속 기능을 수행하는 것이다.

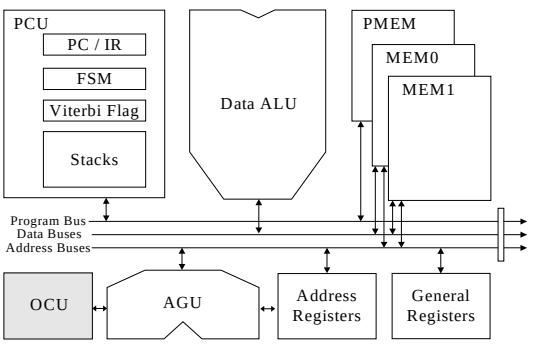


그림 6. 제안된 DSP 구조

그림 6에 나타난 제안된 전체 DSP 구조는 크게 AGU, 프로그램 제어 유닛(Program Control Unit, PCU), DPU, OCU, 프로그램 메모리, 데이터 메모리로 구성되어 있다. DPU 내에 ALU는 경로 매트릭 값의 빠른 비교 연산을 지원하며, 버터플라이 연산 시에 동일한 값을 메모리의 읽기 과정 없이 두 번 사용하기 위해 덧셈기 앞에 전용의 1 사이클 지연 레지스터가 존재한다. OCU는 AGU의 베이스 주소 레지스터와 함께 동작한다.

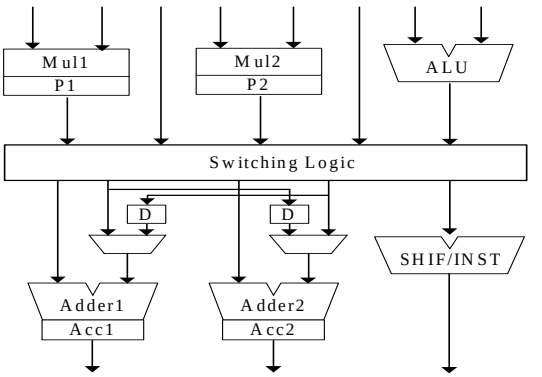


그림 7. 제안된 DPU 구조

그림 7은 제안된 DSP의 DPU 구조를 나타낸다. 제안된 DPU 구조는 두개의 MAC 유닛과 하나의 ALU, 하나의 SHIF/INST(Shift/Insert) 유닛으로 구성되며, 연산기 사이의 스위칭 로직을 사용하여 효과적으로 트랠리스 버터플라이 연산을 수행한다. 즉, 곱셈기와 덧셈기 사이의 스위칭 로직은 각 입력 데이터가 어떠한 누적기(Acc)에도 자유롭게 저장될 수 있도록 한다. DPU는 비터비 연산 시에 16개의 일반 레지스터를 사용하며 빠른 ACS 연산을 위해 전용의 1 사이클 지연 레지스터가 2개 존재한다.

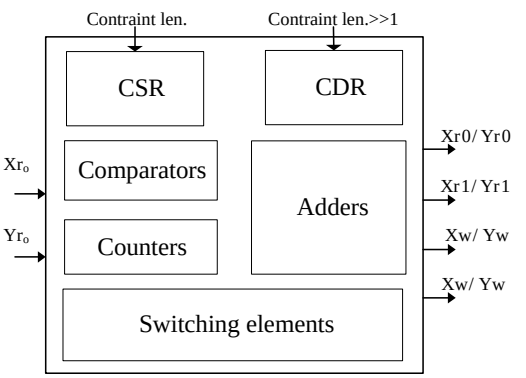


그림 8. 제안된 오프셋 계산 유닛(OCU) 구조

그림 8에 나타난 비터비 전용 OCU는 카운터와 덧셈기, 비교기, 레지스터들 등으로 구성되어 있다. OCU는 비터비 복호 연산 시에 필요한 경로 매트릭 값, 가지 매트릭 값, 경로 결정값(Decision value)의 메모리 읽기, 쓰기 주소 오프셋 값을 생성한다. 또, PCU 내의 비터비 플래그는 OCU의 사용 유무를 결정하며, VS #N과 같은 형식으로 설정된다. 트랠리스 버터플라이를 위한 입력 데이터의 오프셋 주소값

은 구속장 K 값에 의해 결정된다. 이와 같은 전용의 OCU 블록은 비터비 연산을 수행하지 않을 경우에 전력 소모를 줄이기 위해 전력이 차단되도록 설계하였다.

OCU가 생성하는 메모리 주소값 구성은 그림 5에 나타나 있으며, 각각의 메모리는 2개의 읽기와 1개의 쓰기가 동시에 가능한 듀얼 포트(Dual-port) 메모리를 사용한다. 이러한 메모리 맵과 메모리 포트 구조의 사용으로 ACS-TB 연산의 병렬 처리가 가능하다. OCU의 출력 주소값은 AGU의 오프셋 주소 데이터로 사용되며, AGU의 베이스 주소값과 함께 실제 주소값을 생성하게 된다. 예를 들어 만약 베이스 주소 값이 r_0이고 오프셋 주소값이 n_0일 경우, 사용되는 실제 주소값은 $r_0 + n_0$ 가 되며, 짝수 상태에 대한 결과값은 데이터 메모리 0(DM0)에 저장되고 홀수 상태에 대한 결과값은 데이터 메모리 1(DM1)에 저장되도록 주소가 생성된다.

Ⅴ. 성능 비교

표 2는 K=5, D=25인 비터비 복호 과정에 대해, 제안된 DSP와 상용 DSP [1, 2, 9, 10] 간의 성능 비교를 나타낸 것이다. Camel DSP와 TI TMS320c55x는 2개의 MAC과 2개의 ALU가 존재하며, StarCore SC140의 경우 4개의 MAC과 4개의 ALU를 가지고 있다. ZSP500은 2개의 MAC과 3개의 ALU로 구성되어 있다. 그러나 제안된 DSP의 경우 2개의 MAC과 1개의 ALU로 구성되어 있어 다른 상용 DSP에 비해 적은 연산 유닛과 면적을 가진다. 표 2에 나타난 α 값은 연산 데이터를 읽고 쓸 때 발생하는 Load, Store, Move 등의 명령어와 버터플라이의 반복 구조로 인한 브랜치(Branch) 명령어 등의 처리에 사용되는 지연 사이클이다. 이 지연 사이클의 크기는 일반적으로 총 버터플라이 연산 사이클의 30% 정도로 나타나게 된다 [5]. 그러나 표 2에서 볼 수 있듯이 OCU를 사용할 경우 ACS와 TB 연산 시에 발생하는 추가적인 지연 사이클 α 값을 제거할 수 있다.

상용 DSP의 연산 과정은 ACS 연산과 TB 연산이 분리되어 실행되지만, 제안된 DSP는 ACS 연산과 TB 연산이 동시에 실행되도록 제안되었다. 이를 위해 제안된 DSP는 전용의 OCU 블록과 최적화 된 DPU 구조를 가지고 있으며, TB 연산의 실행 사이클을 효율적으로 줄이며 동작하게 된다.

연산 지연 사이클 α 값이 총 버터플라이 연산의 30% 정도이므로 [12], 구속장 K=5의 경우 제안된 DSP가 Carmel DSP 보다 연산 사이클의 관점에서 약 17% 정도의 성능 향상을 보였으며, TMS320c55x 보다는 약 45% 정도의 성능 향상을 확인할 수 있었다. 뿐만 아니라 TB 연산 사이클 관점에서는 ZSP500 보다 약 85%, Starcore SC140 보다 약 90% 정도의 성능 향상이 있었다.

제안된 OCU 블록은 VHDL을 이용하여 구현하였고, Synopsys® Design Compiler를 이용하여 삼성 SEC 0.18 ㎛ 라이브러리로 로직 합성을 수행하였다. 구현된 OCU 블록은 1,460 게이트 개수를 가지며, 최대 지연 시간은 5.75 ㎱이다. 아래의 그림 9는 제안된 OCU 블록의 시뮬레이션 결과를 나타내었다.

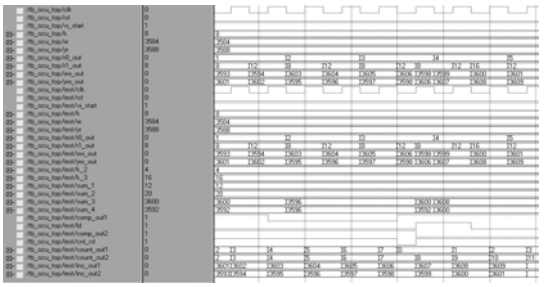


그림 9. 제안된 오프셋 계산 유닛(OCU)의 시뮬레이션 파형

표 2. DSP의 성능 비교(K=5, rate=1/2, D=25)

DSPs / Operations	Starcore (SC140) [1]	Carmel DSP [2]	TMS320 C55x [9]	ZSP500 [10]	제안된 구조
TB 사이클	100+ α	200+ α	95+ α	338+ α	15
ACS 사이클	300+ α	53+ α	400+ α	400+ α	400
ACS+TB 사이클	400+ α	253+ α	495+ α	738+ α	415
MAC 유닛 개수	2	4	2	2	2
ALU 유닛 개수	3	4	2	2	1
DSP 구조	VLIW	CLIW	SIMD	Superscalar	SIMD

α : 추가 지연 사이클 (전체 버터플라이 연산 사이클의 약 30%)

Ⅵ. 결 론

본 논문은 비터비 알고리즘을 위한 효율적인 명령어 집합과 전용의 하드웨어 구조를 제안한다. 제안된 ACSP 명령어와 TBP 명령어는 최적화된 DPU 구조와 함께 DSP에서 ACS 연산과 TB 연산을 병렬적으로 처리할 수 있도록 제안되었다. 또한, 전용의 OCU 블록이 비터비 연산에 필요한 주소의 오프셋 값을 자동으로 생성하여 주므로, 사용자 중심에서 간편하게 비터비 알고리즘을 고속 동작 시킬 수 있다. 제안된 DSP는 VLIW, CLIW(Configurable Long-Instruction Word) 혹은 슈퍼스칼라에 비해 적은 연산기 개수를 가지지만 동일하거나 더욱 향상된 성능을 보였다. 이와 같이 본 논문에서 제안된 무선 통신 표준을 위한 DSP 구조는 적은 하드웨어 복잡도 및 고성능 연산을 특징으로 하므로 다양한 무선 통신 표준과 통신용 SDR 플랫폼에 적용될 수 있다.

참 고 문 헌

[1] StarCore, *How to implement a Viterbi Decoder on the Starcore SC140*, Available: http://www.starcore-dsp.com.

[2] Infineon Technology, *Implementaion of Viterbi Algorithm on CARMEL DSP*, Available: http://www.carmeldsp.com

[3] G. Forney, "Convolutional Codes Maximum Likelihood Decoding," *IEEE Trans. Inform. Theory*, Vol. 25, pp. 222-266, July 1974.

[4] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260-269, Apr.1967.

[5] Jung Hoo Lee, Jae Sung Lee, Myung H. Sunwoo, and Kyung Ho Kim, "Design of New DSP Instructions and Hardware Architecture for The Viterbi Decoding Algorithm," in *Proc. IEEE Int, Symp. Circuits Syst.*, May 2002.

[6] J. Glossner, D, Iancu, J. Lu, E. Hokenek, and M. Moudgill, "A software-defined communications baseband design," *IEEE Commun. Mag.*, vol. 41 Issue:1, Jan. 2003, pp. 120-128.

[7] Nur Engin and Kees van Berkel, "Viterbi Decoding on a Co-processor Architecture with Vector Parallelism," in *Proc. IEEE Workshop on Signal Processing Syst.*, Aug. 2003.

[8] M. Hosemann, R. Habendorf, and G. P. Fettweis, "Hardware-Software Codesign of a 14.4Mbit-64 state-Viterbi Decoder for an Application-specific Digital Signal Processor," in *Proc. IEEE Workshop on Signal Processing Syst.*, Aug. 2003.

[9] Texas Instruments Inc., *TMS320C55x DSP Technical Overview*, Available: http://www.ti.com.

[10] Danny Wilson, *An Efficient Viterbi Decoder Implementation for the ZSP500 DSP Core*, Available: http://www.zsp.com/downloads/

[11] Josef Hausner, *Intergrated Circuit For Next Generation Wireless Communication*, Available: http://www.Infine-on.com.

[12] O. B. Shev, W. Gideon, and B. Eran, *the OakDSP-Core's Viterbi accelerator speeds up digital communications*, Available: http//www.dspg.com

[13] Yingtao Jiang, Yiyan Tang, Yuke Wang, and Swamy M. N. S., "A trace-back-free Viterbi decoder using a new survival path management algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 1, May 2002, pp. 261-234.

[14] J. G. Baek, S. H. Yoon, and J. W. Chong, "Memory efficient pipelined Viterbi decoder with look-ahead trace back," in *Proc. IEEE Int. Conf. Electron. Circuits and Syst.*, vol. 2, 2001, pp. 769-772.

[15] G. Fettweis, "Algebraic Surval Memory Management Design For Viterbi Detectors," *IEEE Trans. Commun.*, vol. 43, pp. 2458-2463, sep. 1995.

박 원 흠(Weon heum Park) 준회원

2001년 2월 강원대학교 전자공학과 졸업(학사)
2003년 2월 아주대학교 전자공학 석사
2003년~현재 삼성전자
<관심분야> 통신용 ASIC 설계

선우 명 훈(Myung hoon Sunwoo) 종신회원

1980년 2월 서강대학교 전자공학과 졸업(학사)
1982년 2월 한국과학기술원 전기 및 전자공학 석사
1982년~1985년 한국전자통신연구소(ETRI) 연구원
1985년~1990년 Univ. of Texas at Austin 전기 및 컴퓨터 공학과 박사
1990년~1992년 미국 Motorola, DSP Chip Operations
1992년~현재 아주대학교 전자공학부 교수
2001년~현재 IEEE Senior Member
2005년~현재 IEEE CAS Society Seoul Chapter Chair
<관심분야> VLSI 및 SoC Architecture, 멀티미디어 통신용 DSP 칩 및 ASIC 설계

오 성 근(Seong keun Oh) 종신회원

1983년 2월 경북대학교 전자공학과 졸업(학사)
1985년 2월 한국과학기술원 전기 및 전자공학 석사
1990년 8월 한국과학기술원 전기 및 전자공학 박사
1993년 12월~1993년 8월 삼성전자(주)
1993년 8월~현재 아주대학교 전자공학과 교수
<관심분야> 이동통신 시스템용 적용 배열 안테나 신호처리 시스템, 이동통신 시스템용 적응등화기, DS-CDMA 수신 시스템, Acoustic Echo Canceller.